

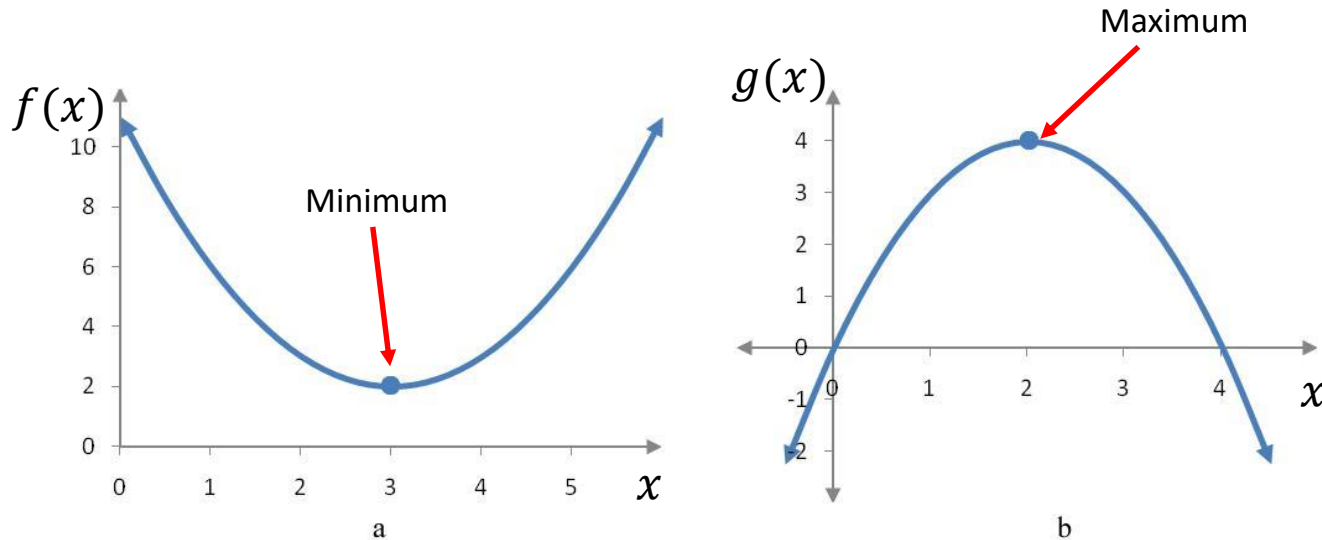
Week 13: Numerical Differentiation and Integration

Acknowledgement: Un-copyrighted photos and pictures were taken from the Internet

Gradient Descent Algorithm

Motivation: Optimization

- **Optimization**: finding the maximum or minimum of a function!
- The **maximum (minimum)** of a function can be found by **differentiation**.
- Many applications in **AI, deep learning**, etc.

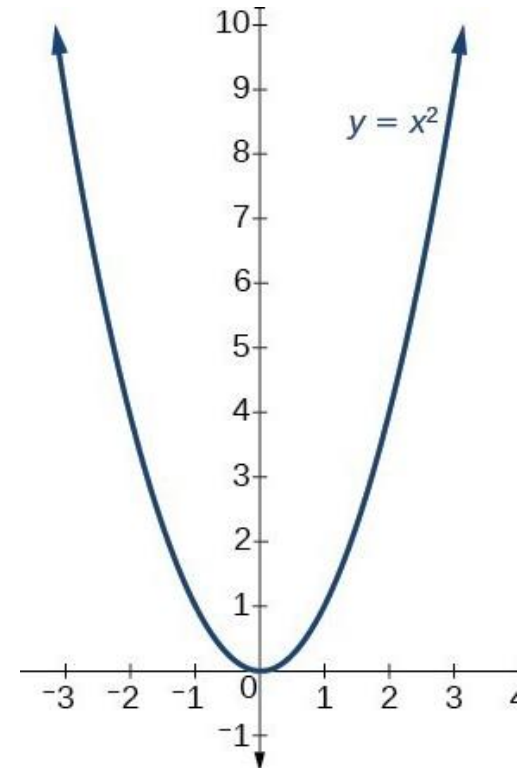


Optimization Problem

- Suppose that you need to find the **minimum** of $f(x) = (x)^2$
- To find it, we find the roots of $f'(x)$.

$$f'(x) = 2(x)$$

- So, the minimum is 0.
- In this case, it is easy to find the root of $2x=0$.



Gradient Descent Algorithm

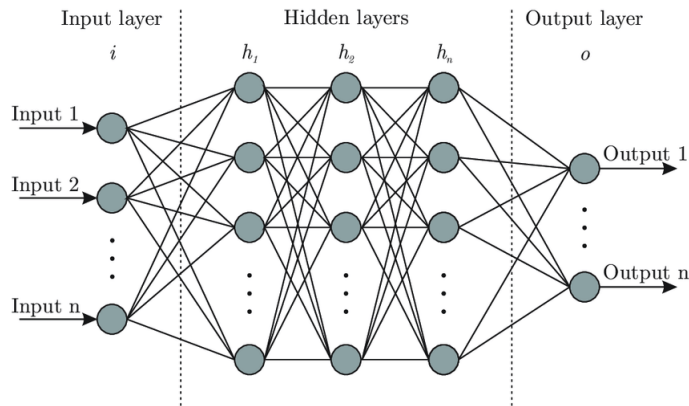
- If $f(x)$ is too complex, we may not be able to find its minimum (maximum) analytically.
- We can use numerical methods to find the minimum or maximum.
- In this lecture, we learn about the gradient descent algorithm.
- It is a numerical algorithm for finding the minima of functions.
- It has many applications in AI and Deep Learning where we deal with complex problems.

Gradient Descent Algorithm

- It can be used to find the minima of **functions** with **multiple variables**.
- Find the values of **x and y** for which the function $f(x, y) = x^2 + y^2$ takes its **smallest value**.
- Find the values of **x, y and z** for which the function $f(x, y, z) = x^2 + y^2 + z^2$ takes its **smallest value**.
- Find the values of **x, y, z and d** for which the function $f(x, y, z, d) = x^2y^2 + z^2 + d^4$ takes its **smallest value**.

Motivation: Neural Networks

- Neural networks have many applications in autonomous driving.



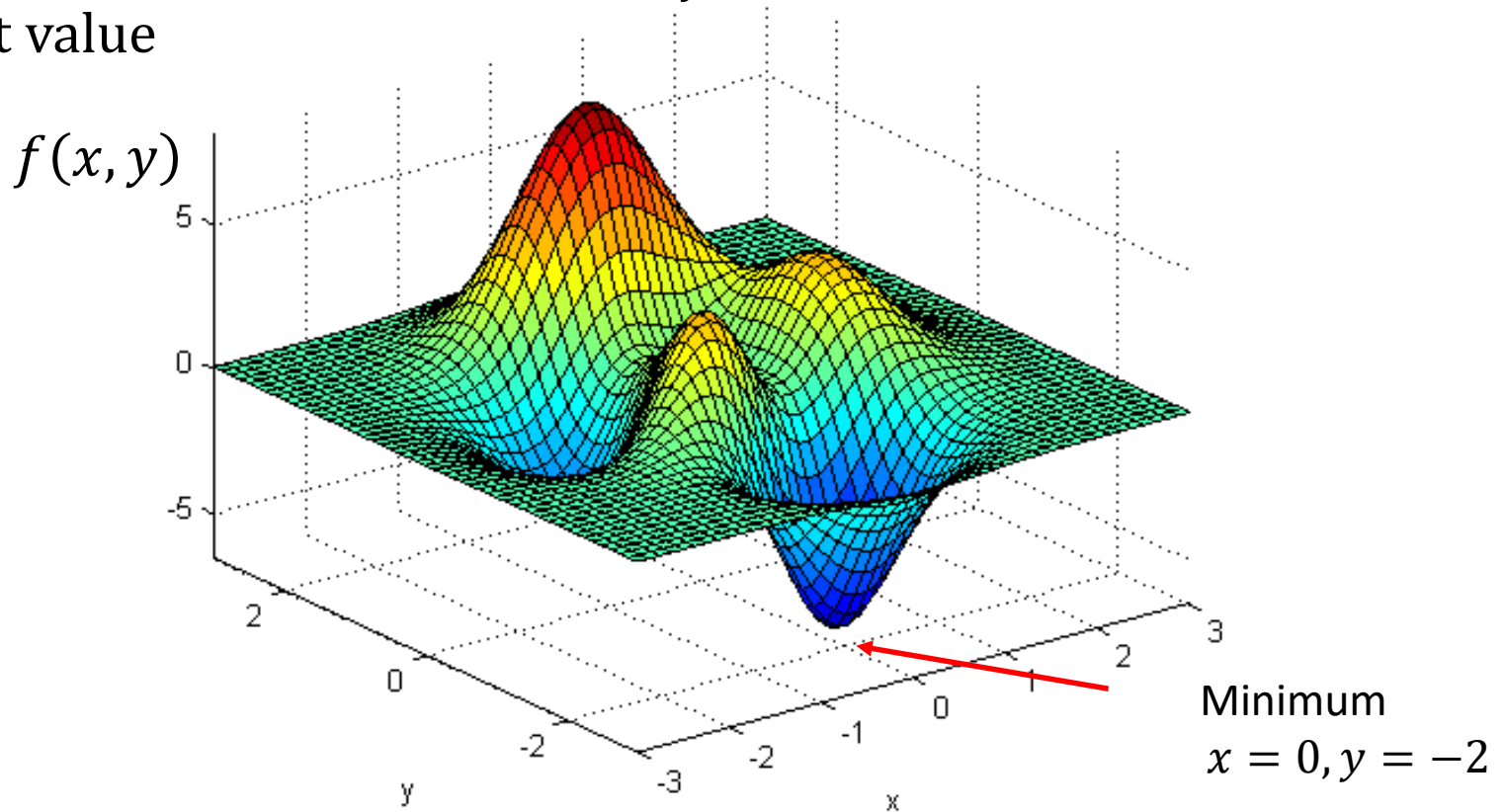
Gradient

- Consider the function $f(x, y)$.
- The **gradient** of $f(x, y)$ at the **point** (x_i, y_i) is a **vector**.
- We **show** the **gradient** of $f(x, y)$ at the point (x_i, y_i) by $\nabla f(x_i, y_i)$.
- The **gradient** of $f(x, y)$ is computed using **its partial derivatives**.

$$\nabla f(x_i, y_i) = \begin{bmatrix} \frac{\partial}{\partial x} f(x_i, y_i) \\ \frac{\partial}{\partial y} f(x_i, y_i) \end{bmatrix}$$

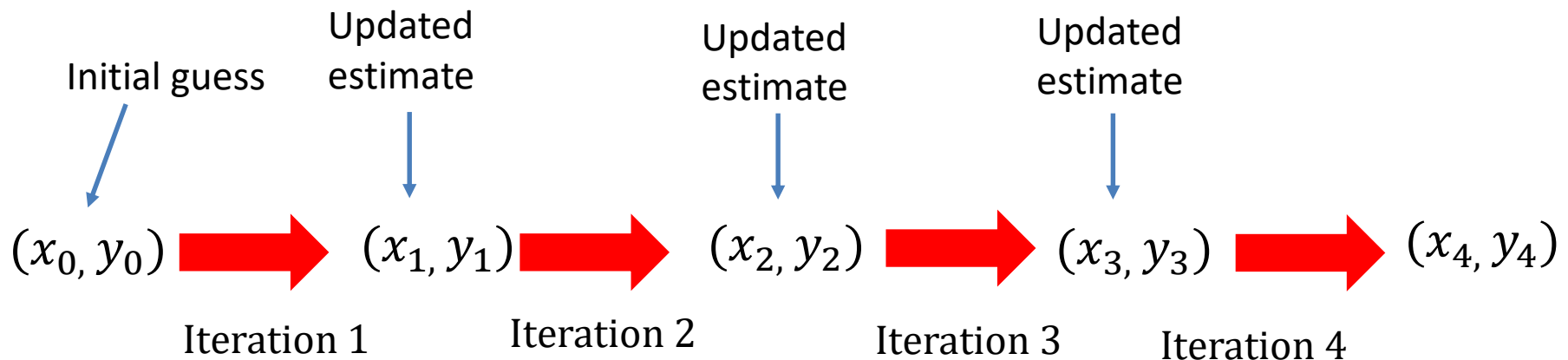
Gradient Descent Algorithm

- Suppose that you need to find the **minimum** of $f(x, y)$.
- We need to find the values of x and y such that the function takes its smallest value



Gradient Descent Algorithm

- Suppose that you need to find the **minimum** of $f(x, y)$.
- In the **gradient descent algorithm**, we start with an **initial point** (x_0, y_0) .
- (x_0, y_0) is the **initial estimate** of the **minimum**.
- At each **iteration**, the **estimate of the minimum** is updated based on **last estimate**.



Gradient Descent Algorithm

- In the gradient descent algorithm, we start with an **initial point** (x_0, y_0) .
- (x_0, y_0) is the **initial estimate** of the **minimum**.
- We then compute the **gradient of $f(x, y)$** at (x_0, y_0) .

$$\nabla f(x_0, y_0) = \begin{bmatrix} \frac{\partial}{\partial x} f(x_0, y_0) \\ \frac{\partial}{\partial y} f(x_0, y_0) \end{bmatrix}$$

Gradient Descent Algorithm

- Next, we update our estimate

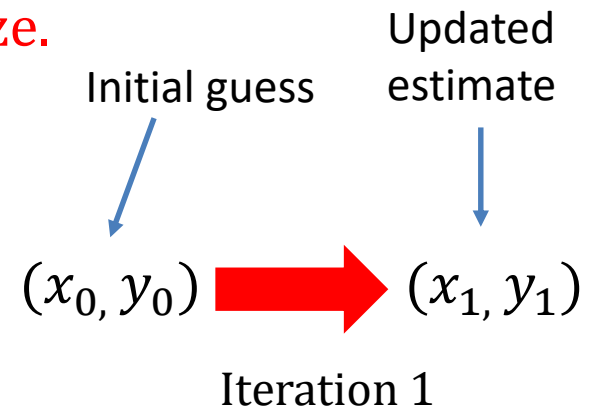
Updated estimate $\rightarrow$ $\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \ominus \mu \nabla f(x_0, y_0)$

Current estimate $\downarrow$ x_0

Step-size $\swarrow$ μ

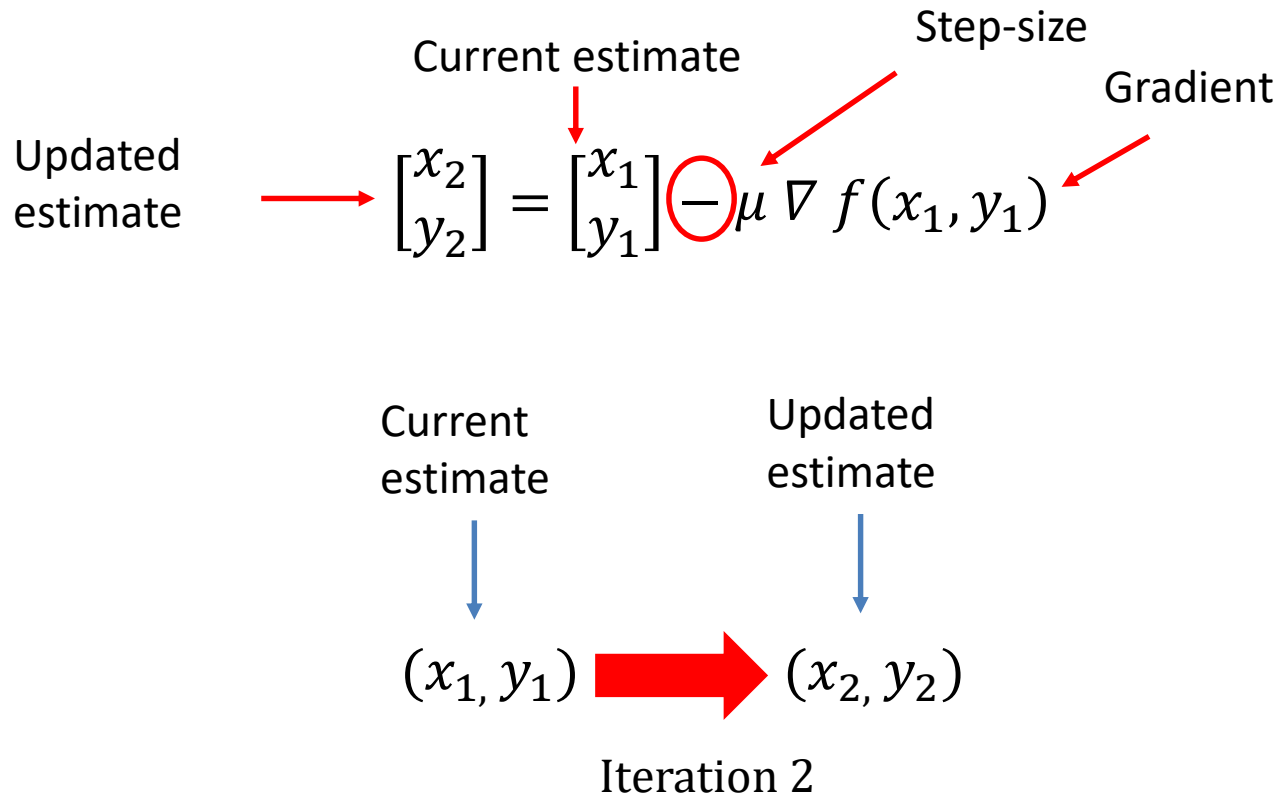
Gradient $\swarrow$ $\nabla f(x_0, y_0)$

- μ is a **small positive number** called the **step-size**.



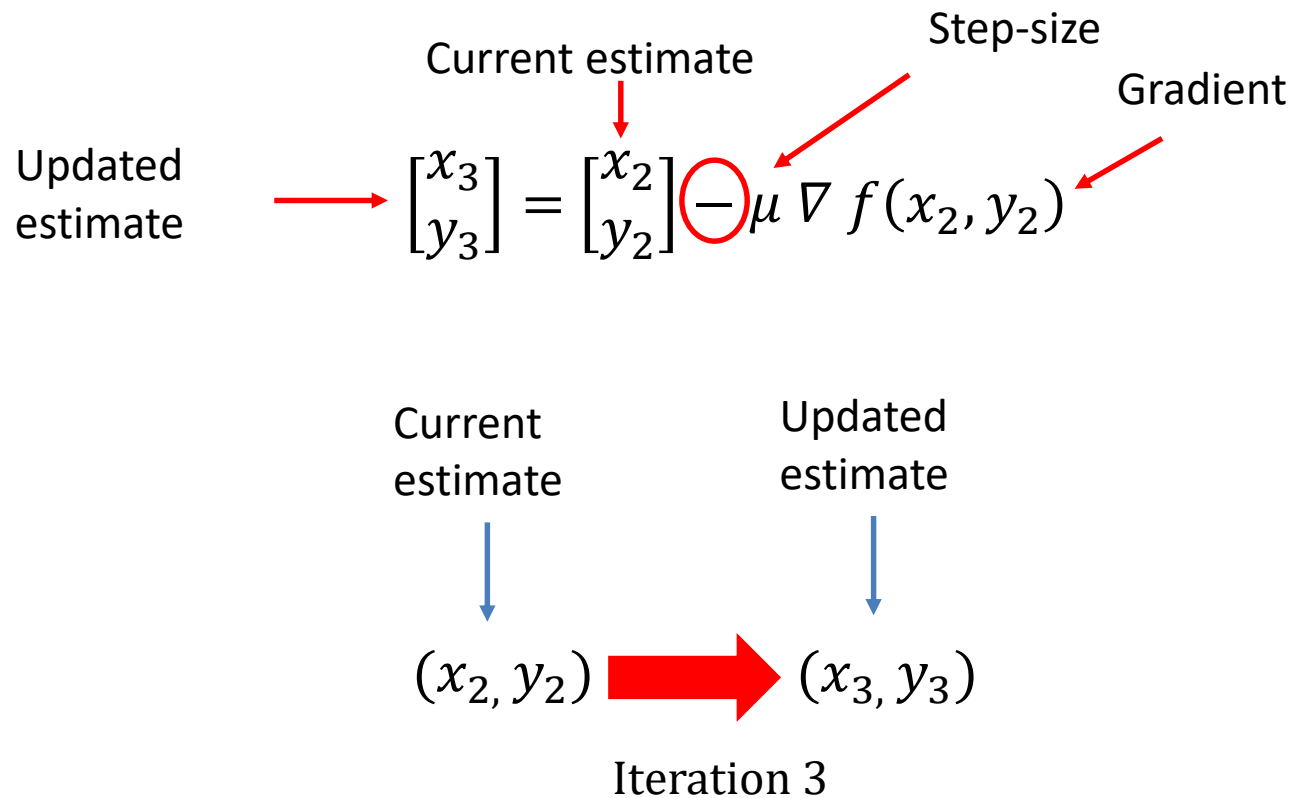
Gradient Descent Algorithm

- In the next iteration, we again update our estimate



Gradient Descent Algorithm

- In the next iteration, we again update our estimate



Gradient Descent Algorithm

- If the current estimate is (x_{k-1}, y_{k-1}) , the updated estimate is computed using

Updated estimate $\longrightarrow$ $\begin{matrix} \text{Current estimate} & & \text{Step-size} & & \text{Gradient} \\ & \downarrow & \swarrow & & \swarrow \\ \begin{bmatrix} x_k \\ y_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \end{bmatrix} \ominus \mu \nabla f(x_{k-1}, y_{k-1}) \end{matrix}$

- We continue until the algorithm converges.

Gradient Descent Algorithm

1. (x_0, y_0) : initial estimate
 2. μ : step-size
 3. $k=1$: iteration counter.
 4. Compute $\nabla f(x_{k-1}, y_{k-1})$
 5. Update the estimate $\begin{bmatrix} x_k \\ y_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \end{bmatrix} - \mu \nabla f(x_{k-1}, y_{k-1})$
 6. Compute the difference vector
$$d_k = \left| \begin{bmatrix} x_k - x_{k-1} \\ y_k - y_{k-1} \end{bmatrix} \right|$$
 7. Stop if all the elements of d_k are less than ϵ .
 8. Otherwise, $k \leftarrow k + 1$ and go to step 4.
- The last generated $\begin{bmatrix} x_k \\ y_k \end{bmatrix}$ is the estimate of the minimum.

Example

- Use the **gradient descent algorithm** to find the **minimum** of $f(x, y) = (x - 2)^2 + (y - 3)^2$.
- $(x_0, y_0) = (0, 0)$
- $\mu = 0.4$
- $\epsilon = 0.001$

Example

- Answer: $f(x, y) = (x - 2)^2 + (y - 3)^2$.
- $\mu = 0.4, \epsilon = 0.001$
- The gradient of $f(x, y)$ is $\nabla f(x, y) = \begin{bmatrix} 2(x - 2) \\ 2(y - 3) \end{bmatrix}$.
- The update rule becomes

$$\begin{bmatrix} x_k \\ y_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \end{bmatrix} - 0.4 \begin{bmatrix} 2(x_{k-1} - 2) \\ 2(y_{k-1} - 3) \end{bmatrix}$$

Example

- After 6 iterations, all the elements of the difference vector are less than ϵ .

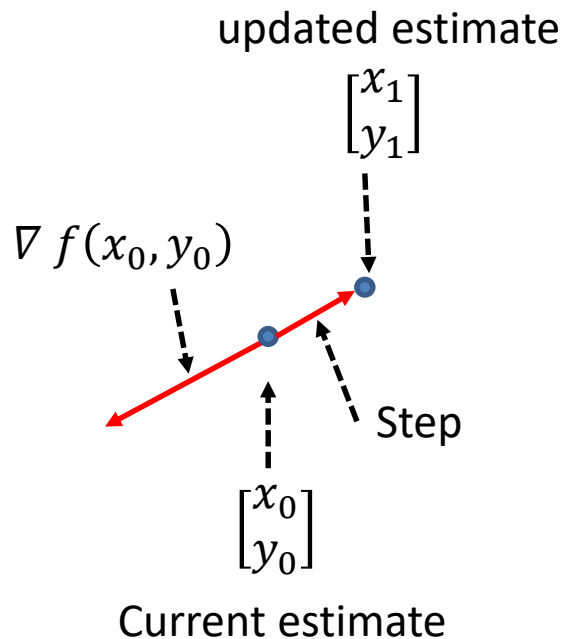
	$k = 0$	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 6$
$\begin{bmatrix} x_k \\ y_k \end{bmatrix}$	0	1.6000	1.9200	1.9840	1.9968	1.9994	1.9999
	0	2.4000	2.8800	2.9760	2.9952	2.9990	2.9998
d_k		1.6000	0.3200	0.0640	0.0128	0.0026	5.1200e-04
		2.4000	0.4800	0.0960	0.0192	0.0038	7.6800e-04

$$\begin{bmatrix} x_k \\ y_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \end{bmatrix} - 0.4 \begin{bmatrix} 2(x_{k-1} - 2) \\ 2(y_{k-1} - 3) \end{bmatrix}$$

- The minimum is located at (1.9999, 2.9998).

How Does it Work?

- We start from (x_0, y_0) .
- Then, we take a **small step** in the **opposite direction** of the **gradient at (x_0, y_0)** . We obtain a new estimate of the minimum (x_1, y_1) .

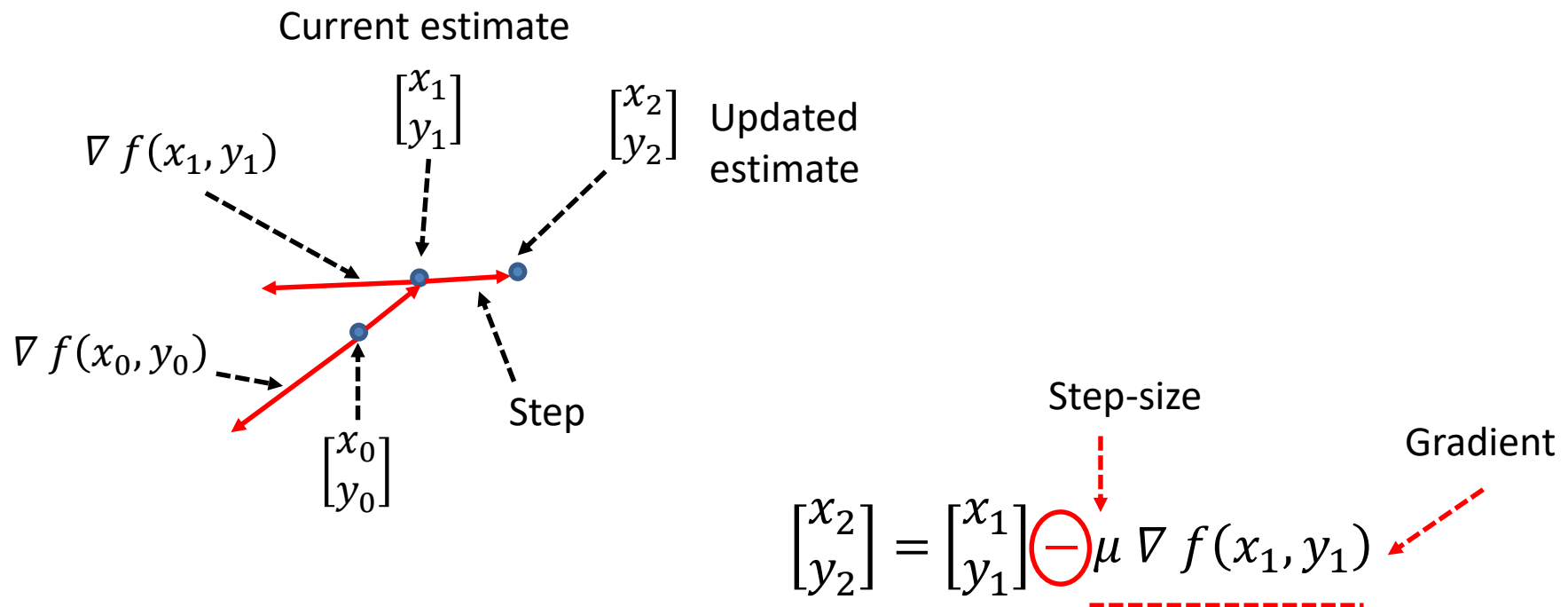


$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \underbrace{\ominus}_{\text{Step-size}} \mu \nabla f(x_0, y_0)$$

Gradient

How Does it Work?

- We take a **small step** in the **opposite direction** of the **gradient at** (x_1, y_1) . We obtain a new estimate of the minimum (x_2, y_2) .



How Does it Work?

- In each iteration, we move in the **opposite direction** of the gradient with a **small step**.

Updated estimate $\rightarrow$ $\begin{bmatrix} x_k \\ y_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \end{bmatrix} \underbrace{-}_{\text{Step-size}} \mu \underbrace{\nabla f(x_{k-1}, y_{k-1})}_{\text{Gradient}}$

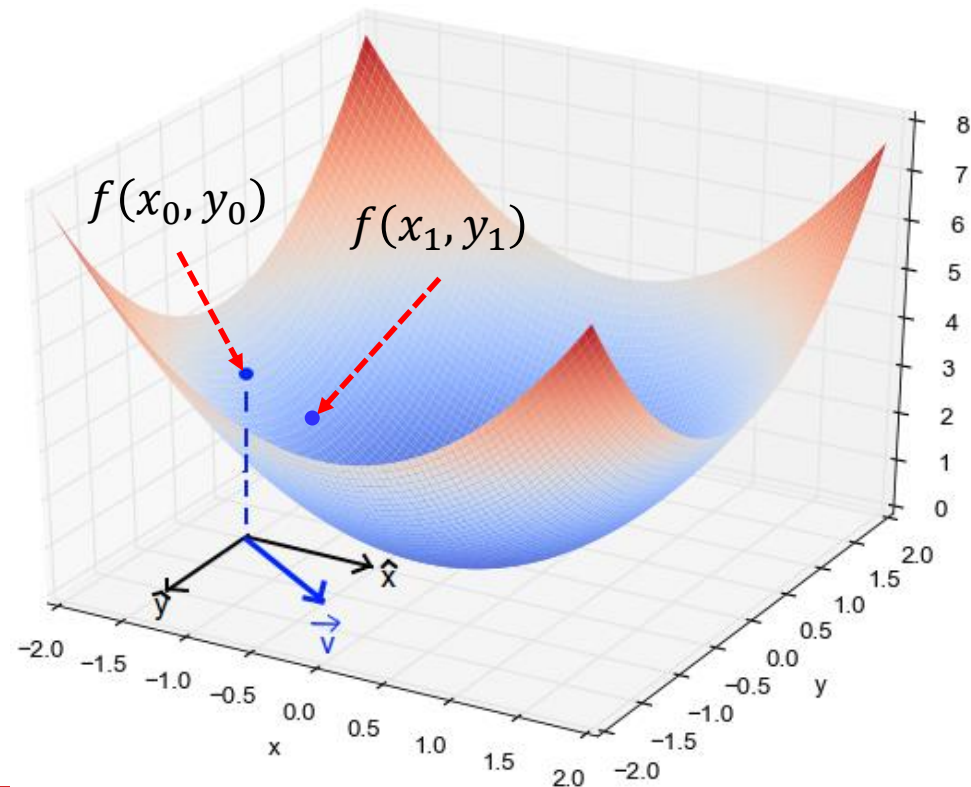
Current estimate

How Does it Work?

- Suppose that we would like to find the **minimum** of the function $f(x, y)$.
- If we start at (x_0, y_0) and take a small step in the **opposite direction of the gradient**, we find a new point.

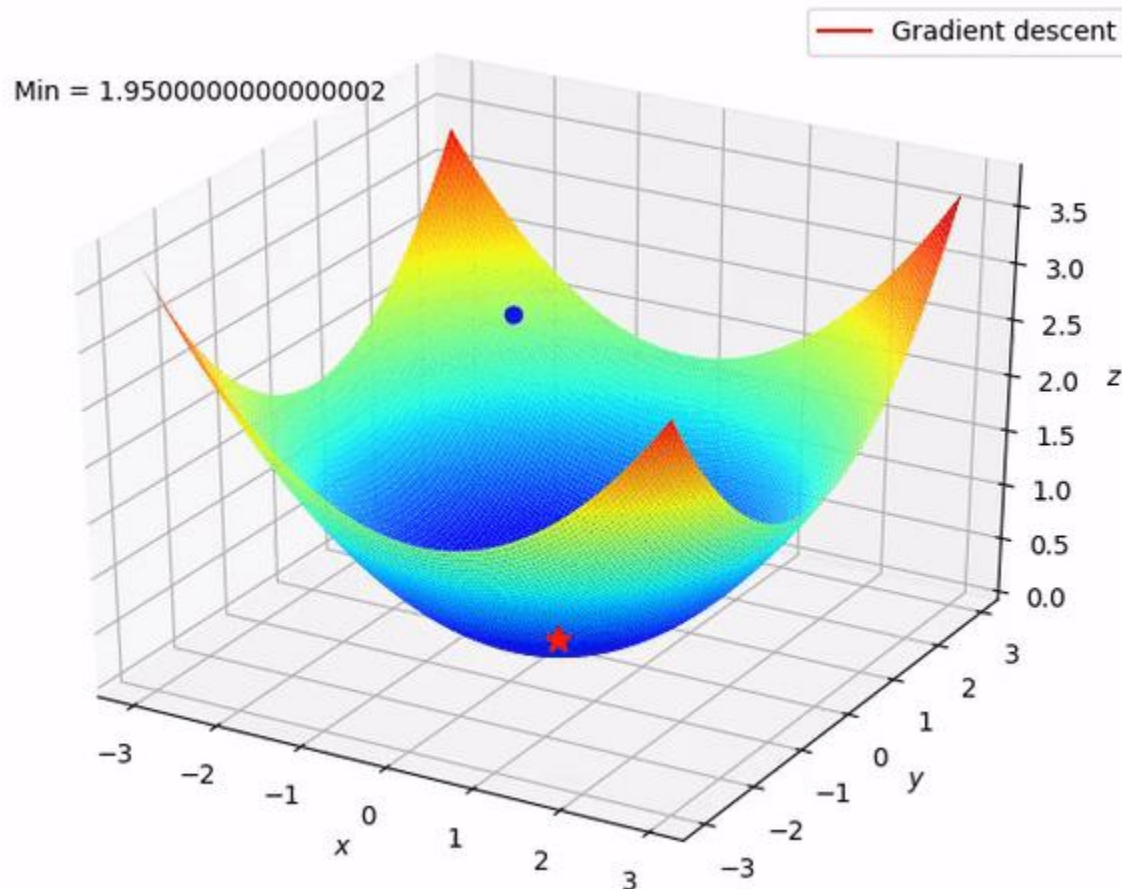
$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} - \mu \nabla f(x_0, y_0)$$

- $f(x_1, y_1)$ will be smaller than $f(x_0, y_0)$.



How Does it Work?

- Each time that we take a **small step** in the **opposite direction** of the **gradient**, the **value of function** at the **new estimate** becomes smaller.



Partial Derivatives

- When the function $f(x, y)$ is complicated, we may not be able to compute its partial derivatives analytically.
- However, we can use numerical methods to estimate the partial derivatives of $f(x, y)$.
- Using the centered difference method, the partial derivatives of $f(x, y)$ around (x_i, y_i) can be estimated as

$$\frac{\partial}{\partial x} f(x_i, y_i) = \frac{f(x_i + h, y_i) - f(x_i - h, y_i)}{2h}$$

$$\frac{\partial}{\partial y} f(x_i, y_i) = \frac{f(x_i, y_i + h) - f(x_i, y_i - h)}{2h}$$

Gradient Descent Algorithm

- The **algorithm** works if the **function** $f(\cdot)$ depends on more than **two variables**.
- At **each step**, we compute the **gradient** at the **current estimate**.
- We then take **a small step** in the **opposite** direction of the **gradient** to obtain the **updated estimate**.
- For example, the update rule for $f(x, y, z)$ is

$$\begin{bmatrix} x_k \\ y_k \\ z_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ z_{k-1} \end{bmatrix} - \mu \nabla f(x_{k-1}, y_{k-1}, z_{k-1})$$

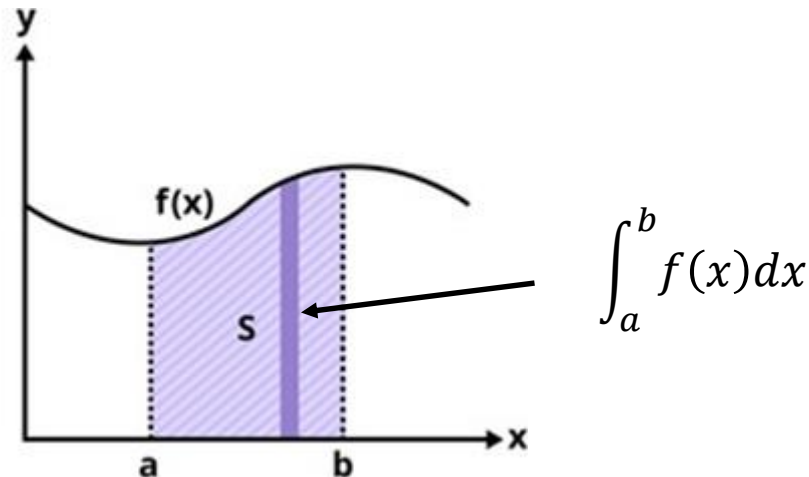
Numerical Integration

Integration

- In **engineering applications**, we often need to **integrate functions**.
- The **integral of $f(x)$** over the **interval $[a, b]$** is denoted by

$$\int_a^b f(x) dx$$

- $\int_a^b f(x) dx$ is equal to the **area under $f(x)$ from a to b** .



Integration

- If function $f(x)$ is relatively **simple**, we can compute its **integral** in **closed-form**.
- For example, when $f(x)$ is a **polynomial**.

- Example:
$$\int_a^b x^p dx = \frac{x^{p+1}}{p+1} \Big|_a^b$$

- For example, we can easily compute the integral of **$\sin(x)$** .

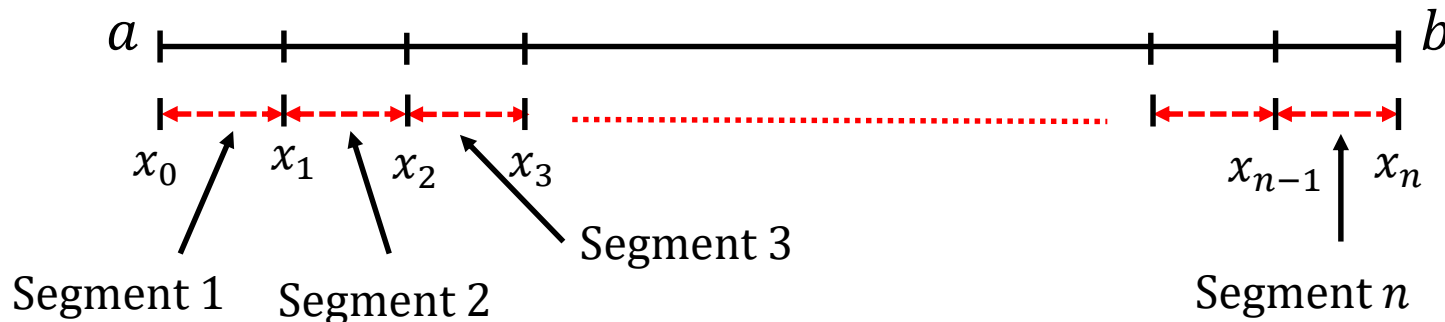
Integration

- When function $f(x)$ is **complicated**, we may not be able to compute its integral in **closed-form**.
- $f(x) = \sin(x^2)$.
- $f(x) = \sin(\log(x))$.
- $f(x) = \frac{\sin(x)}{x}$.
- In such cases, we need **numerical methods** to compute the **integrals**.
- In this lecture, we learn two methods for numerical integration:
 1. **Trapezoidal rule**
 2. **Simpson's 1/3 Rule**

Trapezoidal rule

Trapezoidal rule

- Suppose that we need to **numerically compute** the **integral** of $f(x)$ over the interval $[a, b]$.
- In **numerical integration**, we divide **the interval $[a, b]$** into **n segments**. Here, **n** is an **integer**.
- The **length of each segment** is $h = \frac{b-a}{n}$.
- This gives us **$n + 1$ base points** : $x_0, x_1, \dots, x_n$



Trapezoidal rule

- We need to find the base points.

- $x_0 = a$

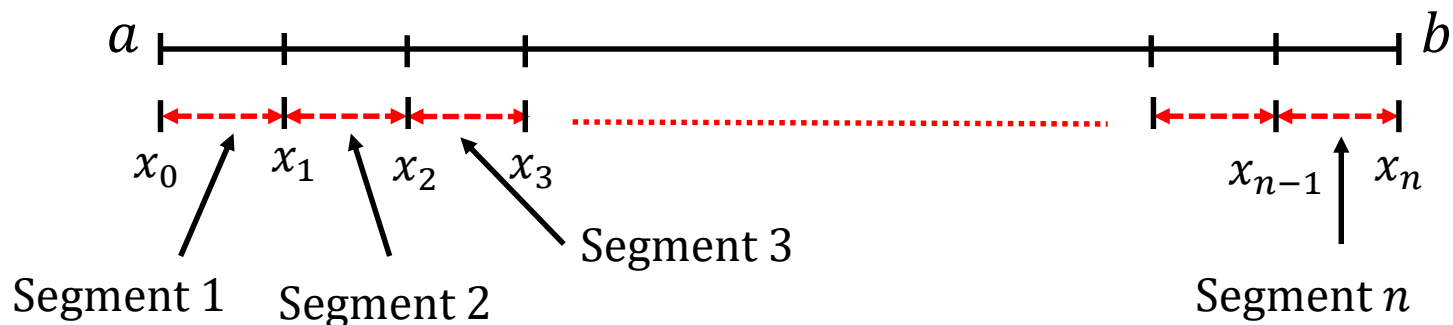
- $x_1 = a + h$

- $x_2 = a + 2h$

$$h = \frac{b - a}{n}$$

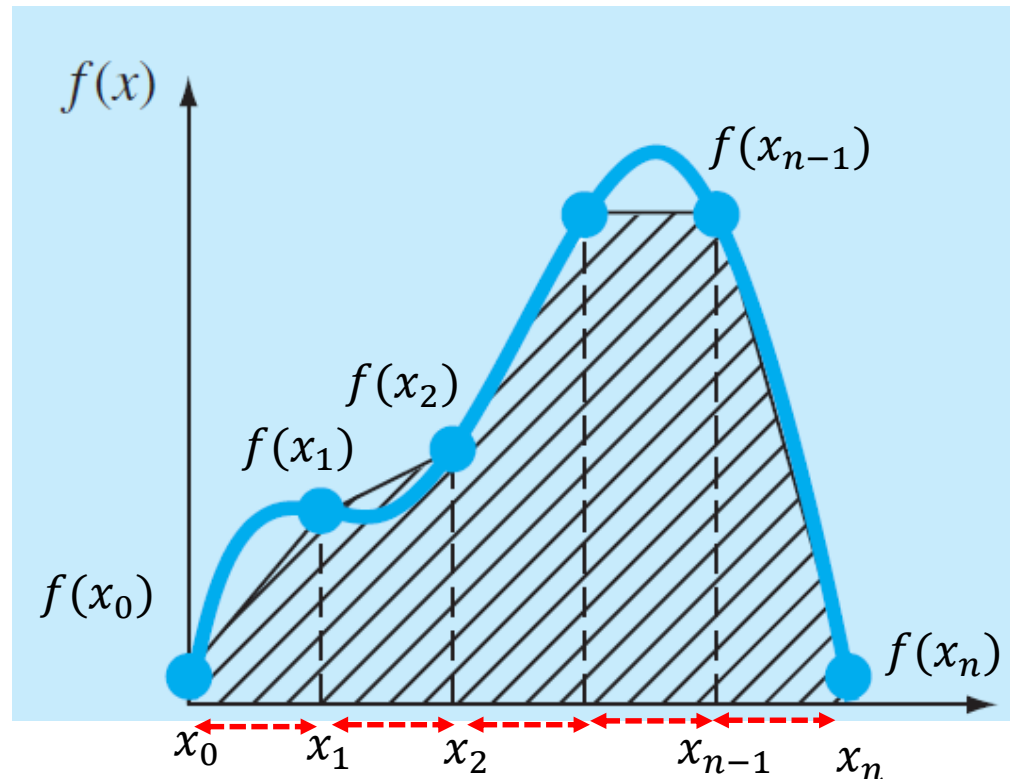
- $x_i = a + ih$

- $x_n = a + nh = b$



Trapezoidal rule

- In the trapezoidal rule, we **first estimate** the area of the function under **each segment**.
- We then **add up** the estimates for segments to obtain the **approximation of integral**.

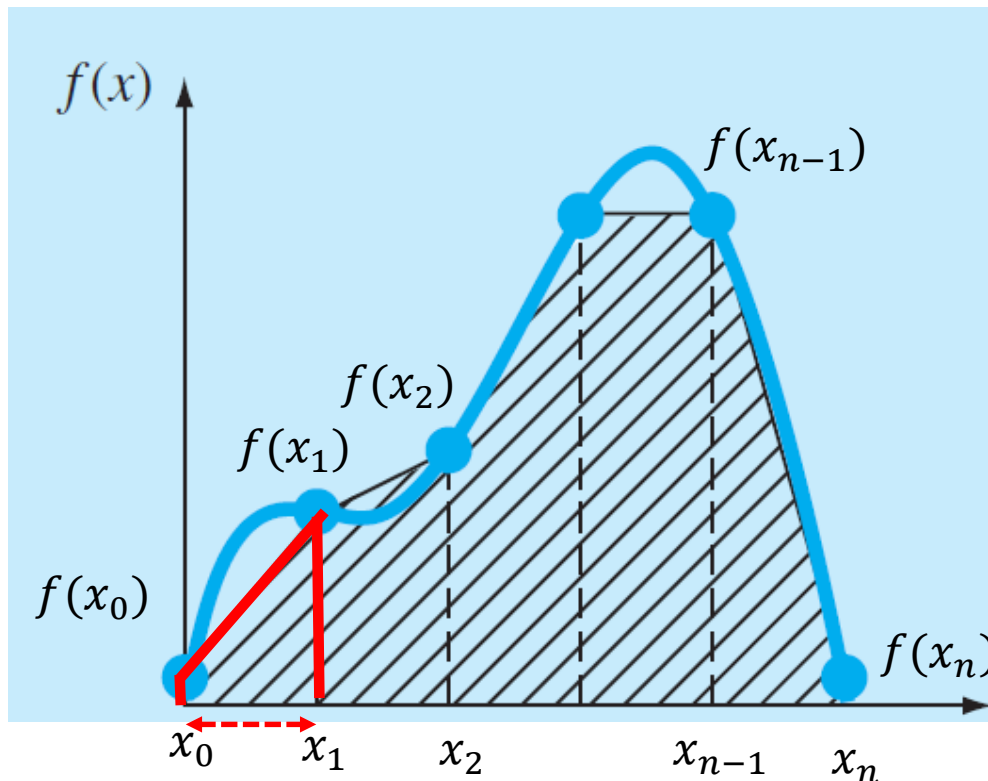


Trapezoidal rule

- The **area** under $f(x)$ from x_0 to x_1 is approximated by .

$$h \frac{f(x_0) + f(x_1)}{2}$$

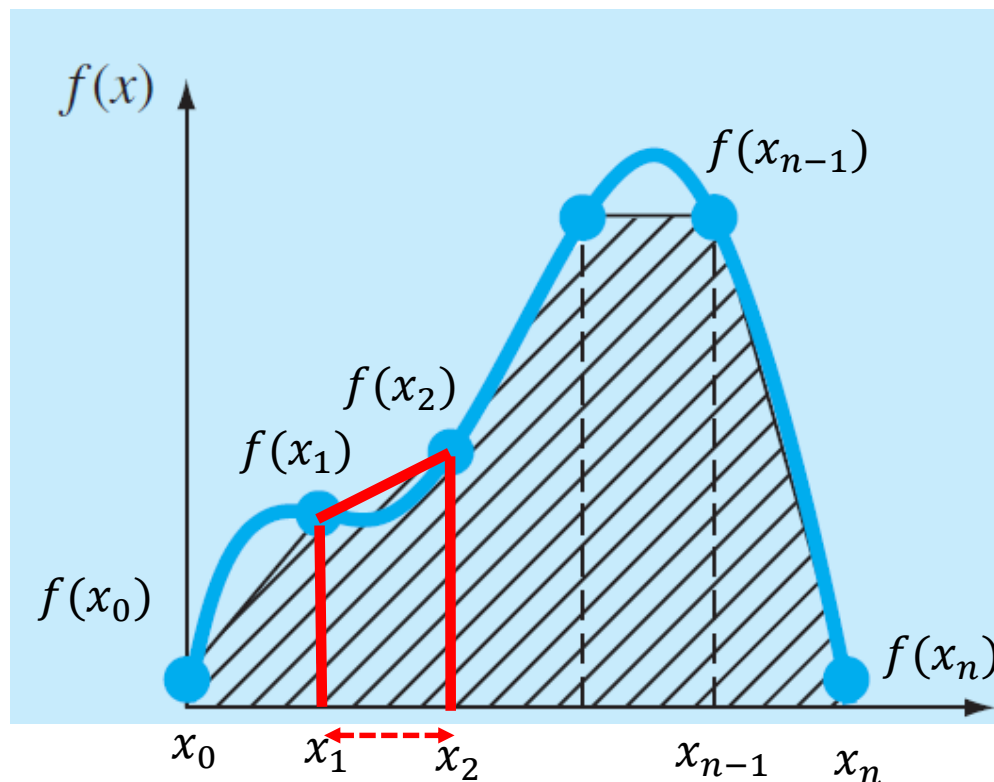
$$h = \frac{b - a}{n}$$



Trapezoidal rule

- The **area** under $f(x)$ from x_1 to x_2 is approximated by .

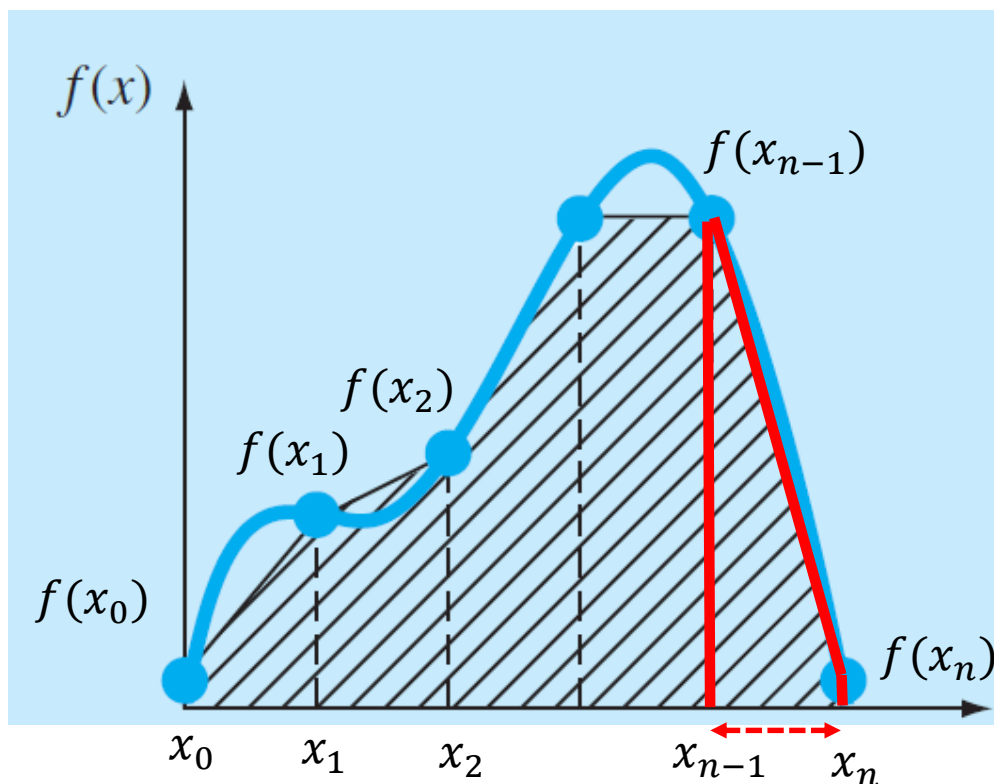
$$h \frac{f(x_1) + f(x_2)}{2}$$



Trapezoidal rule

- The **area** under $f(x)$ from x_{n-1} to x_n is approximated by .

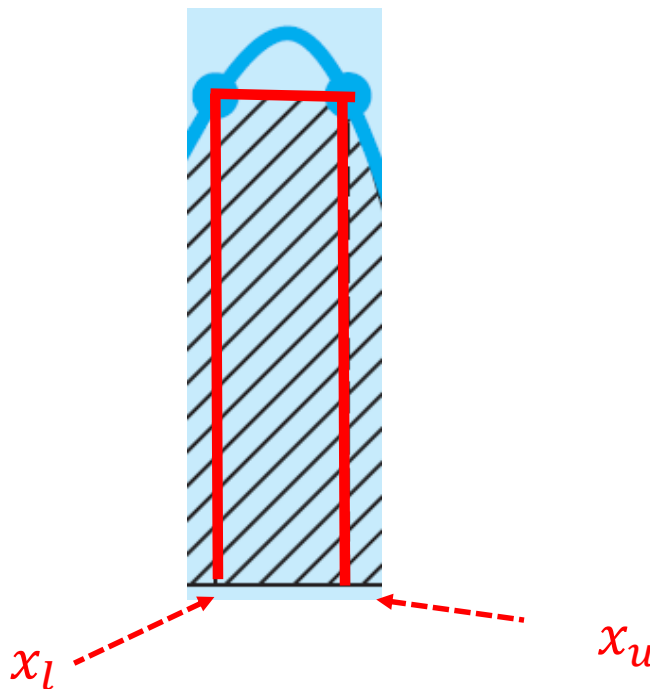
$$h \frac{f(x_{n-1}) + f(x_n)}{2}$$



Trapezoidal rule

- The **area** under $f(x)$ from over the segment x_l to x_u is approximated by

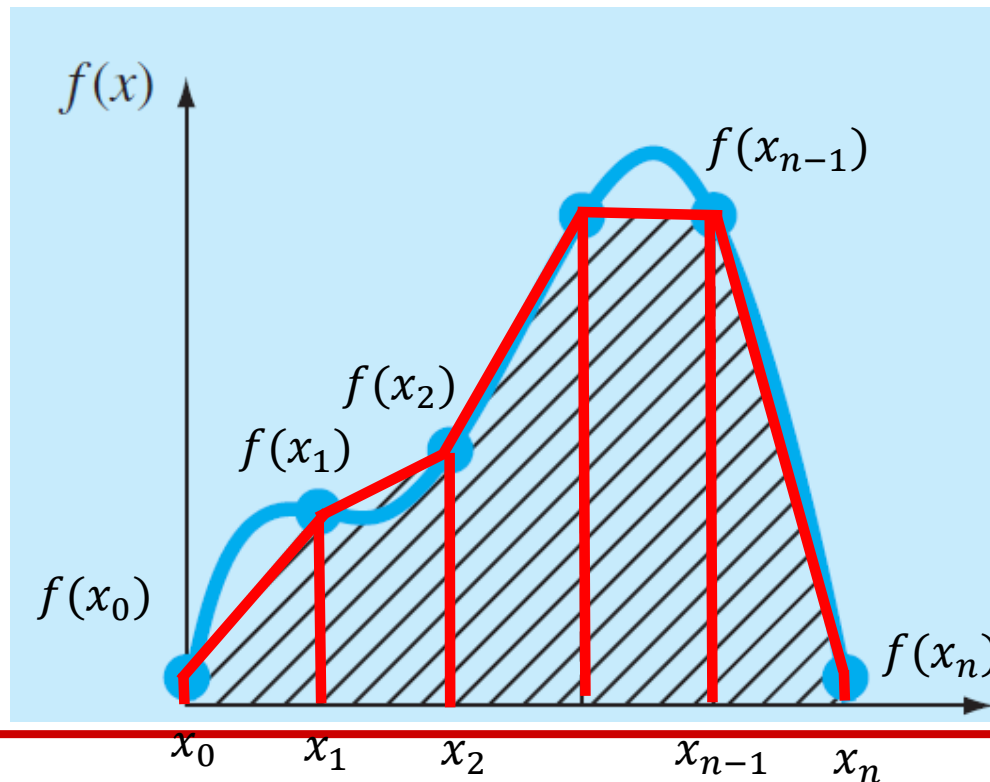
$$Area = h \frac{f(x_l) + f(x_u)}{2}$$



Trapezoidal rule

- To **compute** the **integral** of $f(x)$, we add up the estimated areas

$$\int_a^b f(x)dx \approx h \frac{f(x_0) + f(x_1)}{2} + h \frac{f(x_1) + f(x_2)}{2} + \cdots + h \frac{f(x_{n-1}) + f(x_n)}{2}$$



Example

Problem Statement. Use the two-segment trapezoidal rule to estimate the integral of

$$f(x) = 0.2 + 25x - 200x^2 + 675x^3 - 900x^4 + 400x^5$$

from $a = 0$ to $b = 0.8$. Recall that the correct value for the integral is 1.640533.

Compute the percentage of the true relative error for your answer.

Trapezoidal rule

- $n = 2, a = 0, b = 0.8.$

- So, $h = 0.4.$

$$h = \frac{b - a}{n}$$

- $x_0 = 0 \quad f(0) = 0.2$

- $x_1 = 0.4 \quad f(0.4) = 2.456$

- $x_2 = 0.8 \quad f(0.8) = 0.232$

Trapezoidal rule

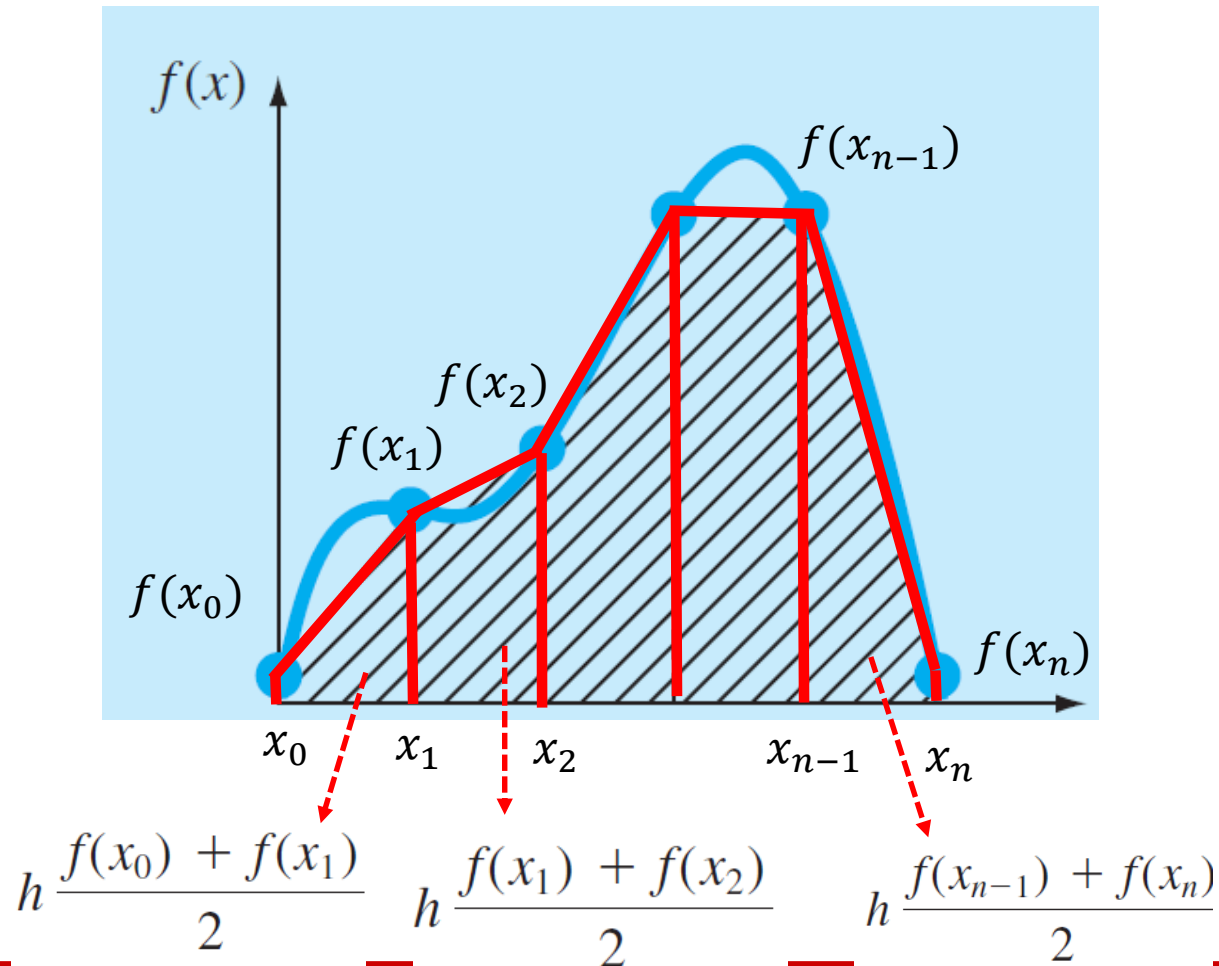
- The approximate value of the integral is

$$\begin{aligned}\int_a^b f(x)dx &\approx h \frac{f(x_0) + f(x_1)}{2} + h \frac{f(x_1) + f(x_2)}{2} \\ &= 0.8 \frac{0.2 + 2(2.456) + 0.232}{4} = 1.0688\end{aligned}$$

- The error is $E_t = 1.640533 - 1.0688 = 0.57173$
- The percentage of the relative error is $\varepsilon_t = 34.9\%$

Trapezoidal rule

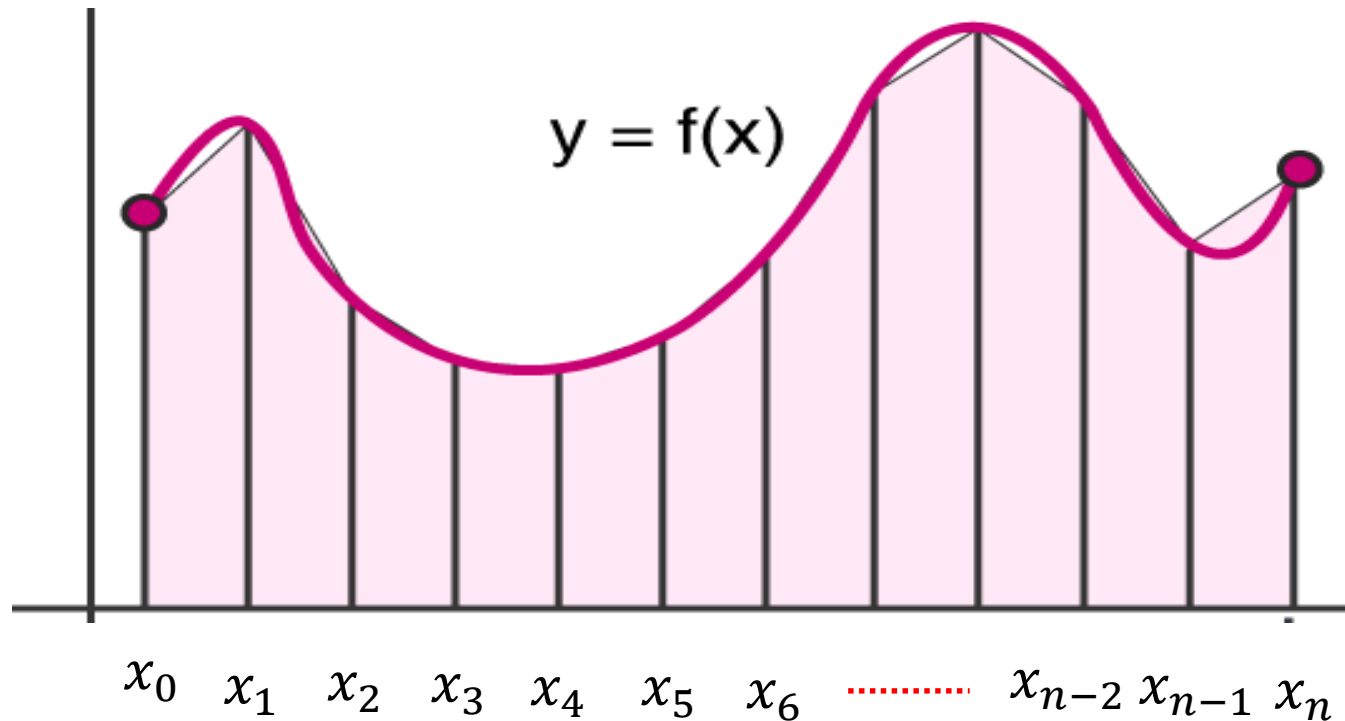
- The **area** under $f(x)$ in each **segment** is approximated by the **area** of the **trapezoid** formed by the **base points** of that **segment**.



Simpson's 1/3 Rule

Simpson's 1/3 Rule

- Suppose that we need to **numerically compute** the **integral** of $f(x)$ over the interval $[a, b]$.
- We first find the base points.



Simpson's 1/3 Rule

- We need to find the base points.

- $x_0 = a$

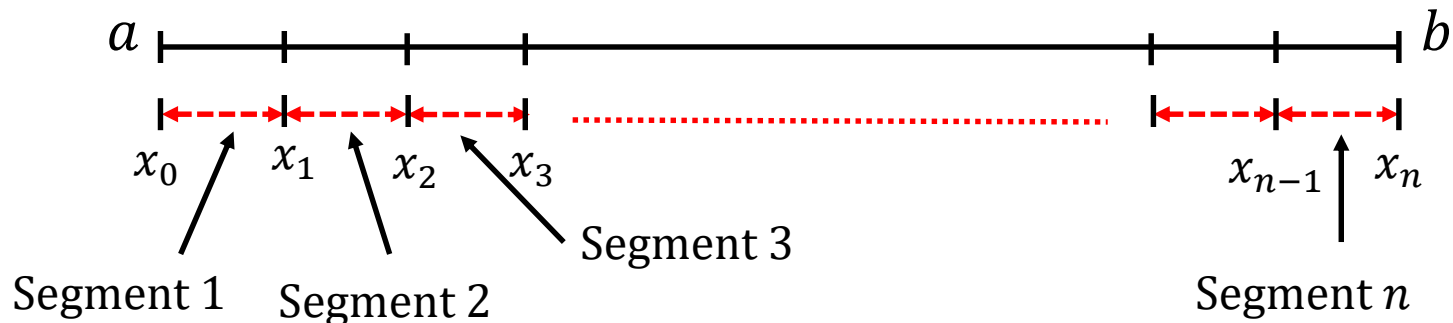
- $x_1 = a + h$

- $x_2 = a + 2h$

$$h = \frac{b - a}{n}$$

- $x_i = a + ih$

- $x_n = a + nh = b$



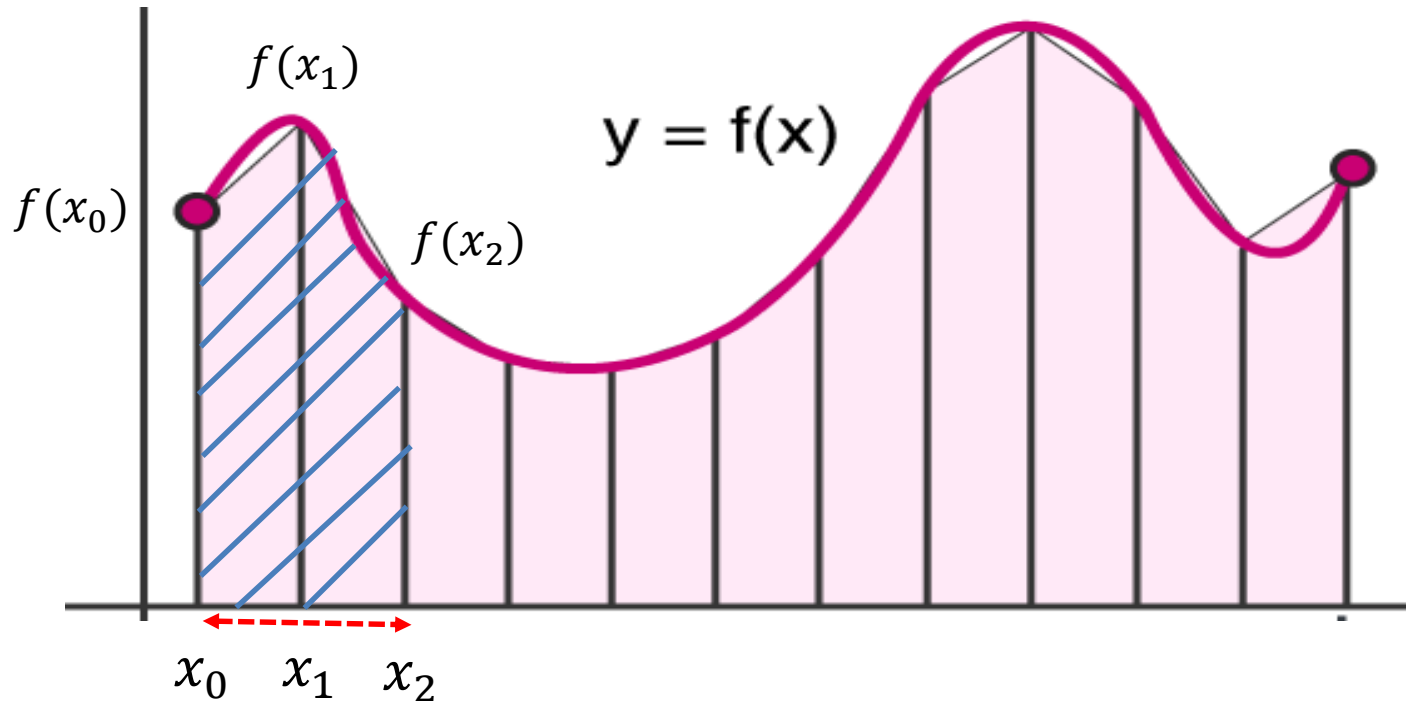
Simpson's 1/3 Rule

- We first approximate the **area under the function** from x_0 to x_2 .
- We then approximate the **area under the function** from x_2 to x_4 .
- We then approximate the **area under the function** from x_4 to x_6 .
- We then approximate the **area under the function** from x_6 to x_8 .
- We continue this procedure.
- Lastly, we approximate the **area under the function** from x_{n-2} to x_n .

Simpson's 1/3 Rule

- We first approximate the **area under the function** from x_0 to x_2 using the formula

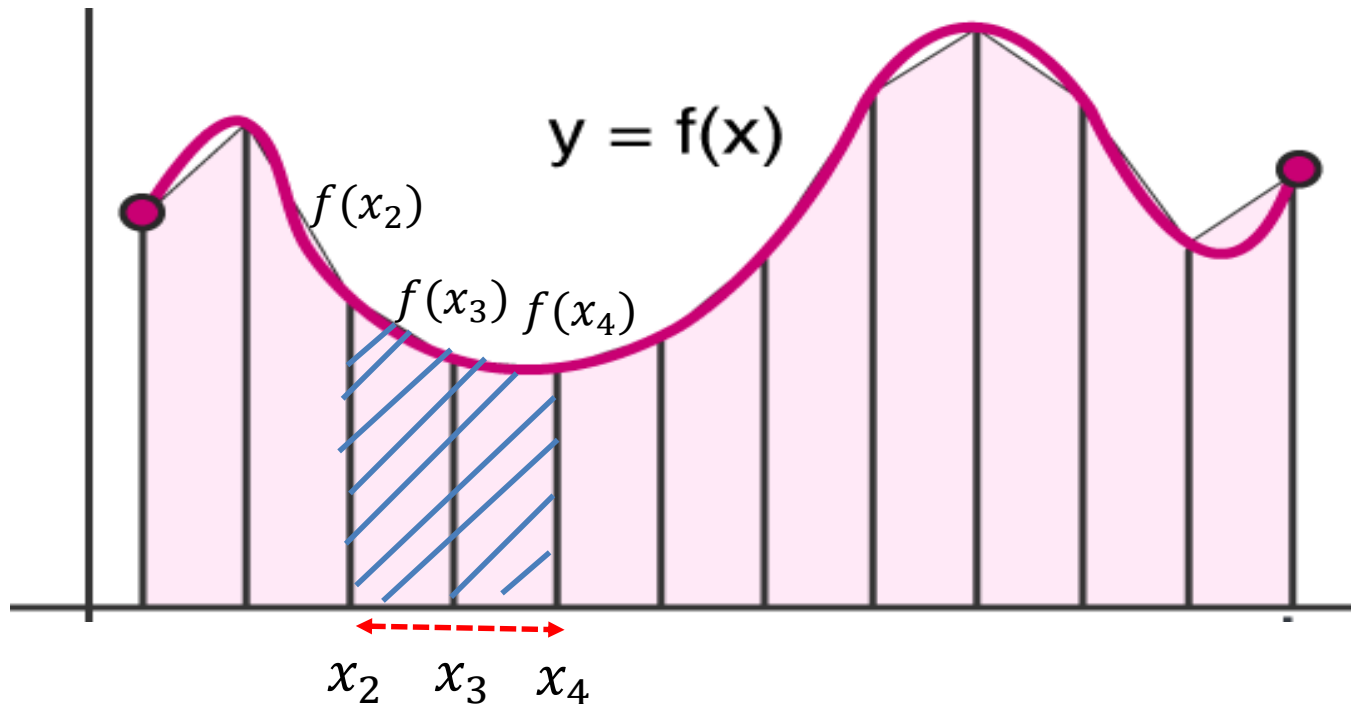
$$2h \frac{f(x_0) + 4f(x_1) + f(x_2)}{6}$$



Simpson's 1/3 Rule

- We next approximate the **area under the function** from x_2 to x_4 using the formula

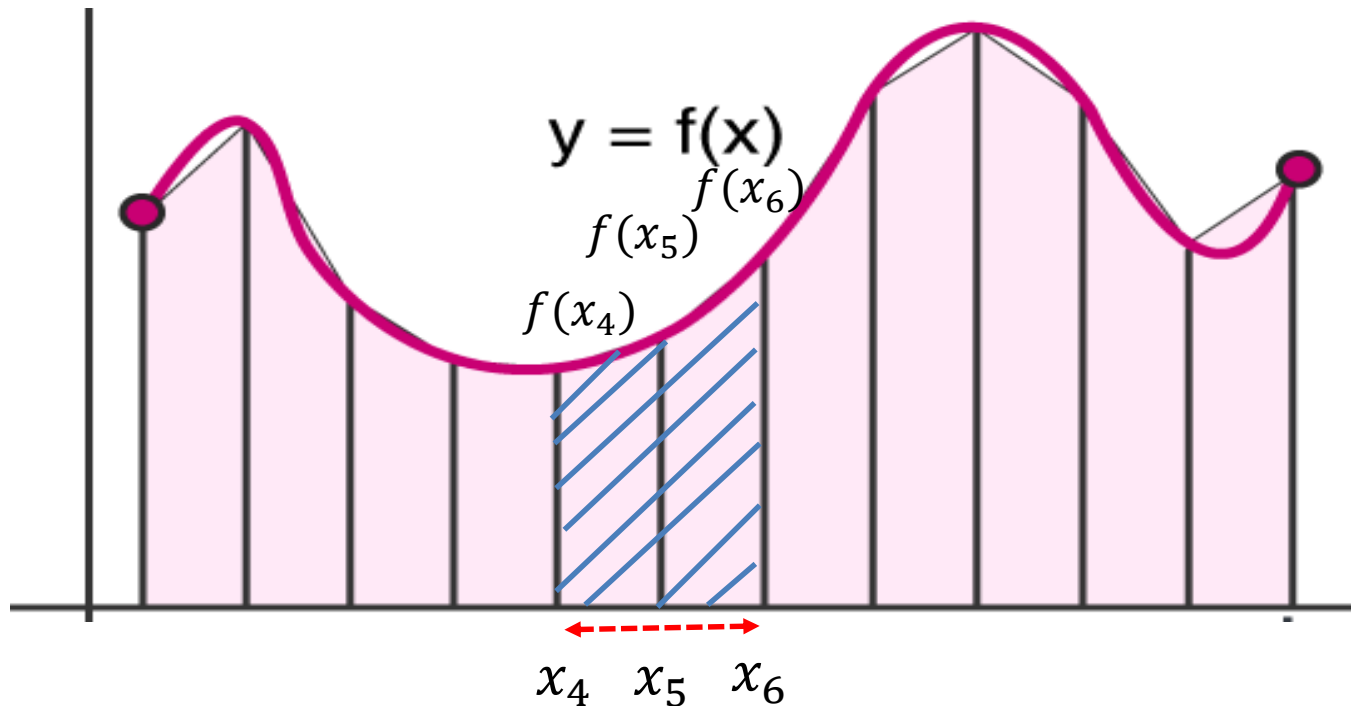
$$2h \frac{f(x_2) + 4f(x_3) + f(x_4)}{6}$$



Simpson's 1/3 Rule

- We next approximate the **area under the function** from x_4 to x_6 using the formula

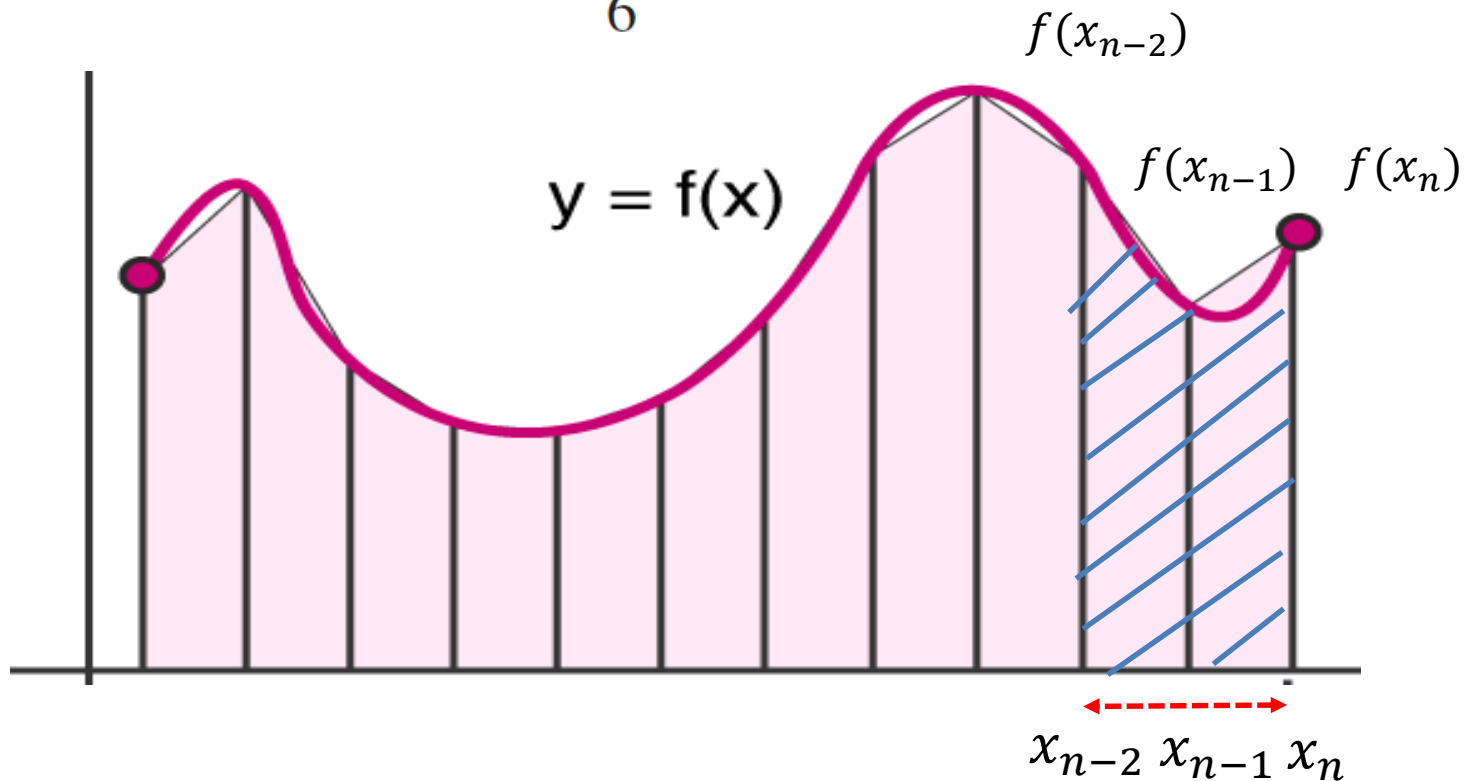
$$2h \frac{f(x_4) + 4f(x_5) + f(x_6)}{6}$$



Simpson's 1/3 Rule

- Lastly, we approximate the **area under the function** from x_{n-2} to x_n using the formula

$$2h \frac{f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)}{6}$$

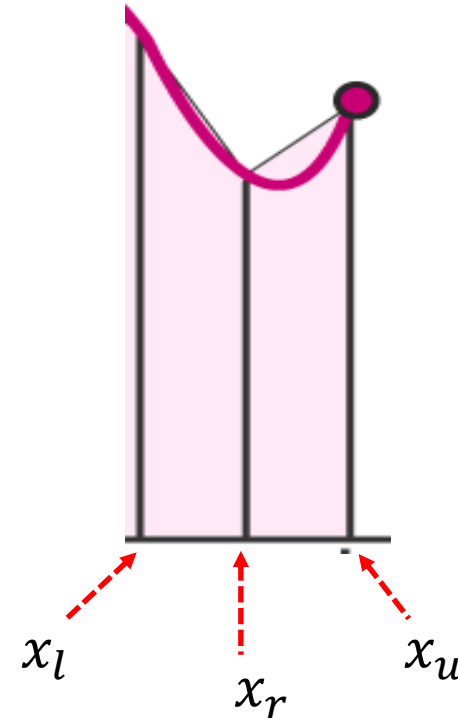


Simpson's 1/3 Rule

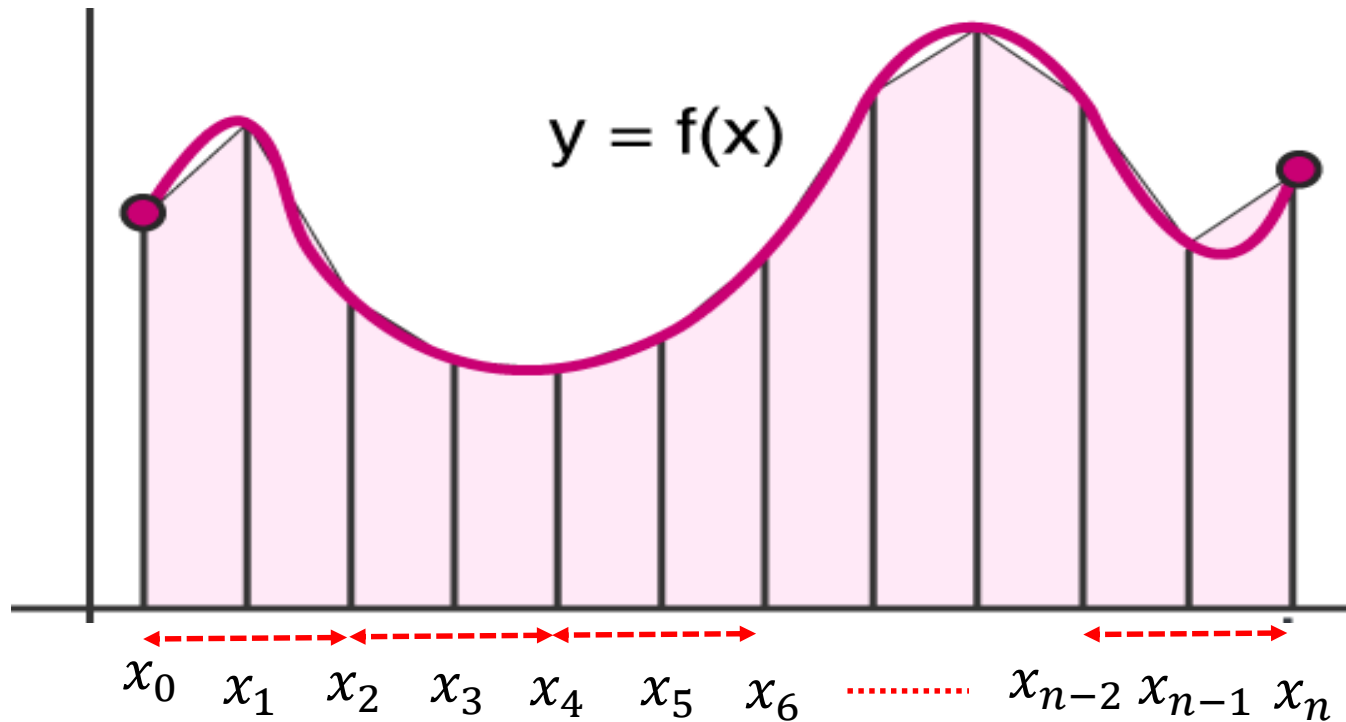
- Use the following **formula** to approximate the area under $f(x)$ from x_l to x_u .

$$\text{Area} = 2h \frac{f(x_l) + 4f(x_r) + f(x_u)}{6}$$

$$x_r = \frac{x_l + x_u}{2}$$



Simpson's 1/3 Rule



- To **compute** the **integral** of $f(x)$, we add up the estimated areas

$$I \cong 2h \frac{f(x_0) + 4f(x_1) + f(x_2)}{6} + 2h \frac{f(x_2) + 4f(x_3) + f(x_4)}{6} \\ + \dots + 2h \frac{f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)}{6}$$

Example

Problem Statement. Estimate the integral of

$$f(x) = 0.2 + 25x - 200x^2 + 675x^3 - 900x^4 + 400x^5$$

from $a = 0$ to $b = 0.8$. Recall that the exact integral is 1.640533.

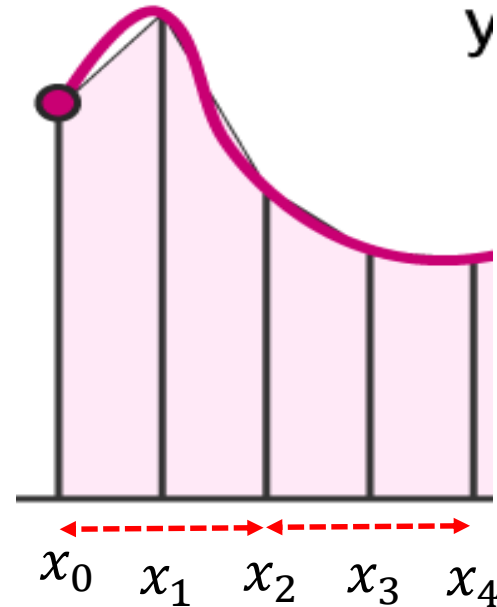
Use $n = 4$ and compute the percentage of the true relative error for your answer.

Simpson's 1/3 Rule

- We have $a = 0, b = 0.8, n = 4$.

- So, $h = 0.2$.

• $x_0 = 0$	$f(0) = 0.2$
• $x_1 = 0.2$	$f(0.2) = 1.288$
• $x_2 = 0.4$	$f(0.4) = 2.456$
• $x_3 = 0.6$	$f(0.6) = 3.464$
• $x_4 = 0.8$	$f(0.8) = 0.232$



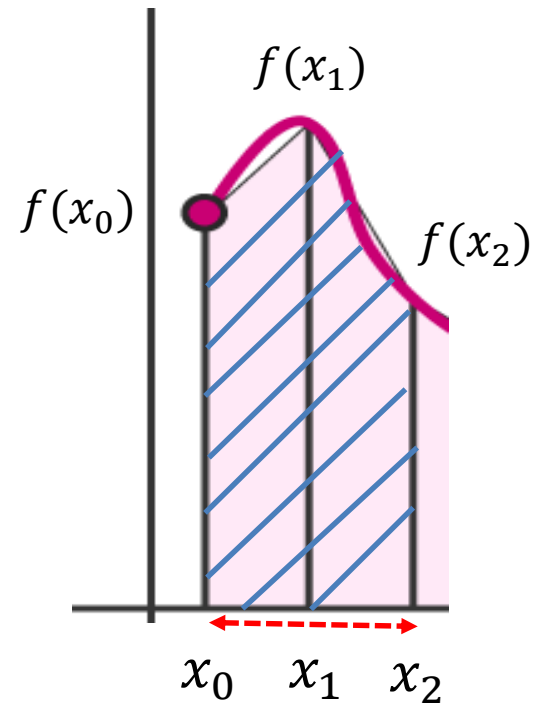
- We need estimate the area from x_0 to x_2 and the area from x_2 to x_4 .

Simpson's 1/3 Rule

- We first estimate the area from x_0 to x_2 .

- $x_0 = 0$ $f(0) = 0.2$
- $x_1 = 0.2$ $f(0.2) = 1.288$
- $x_2 = 0.4$ $f(0.4) = 2.456$

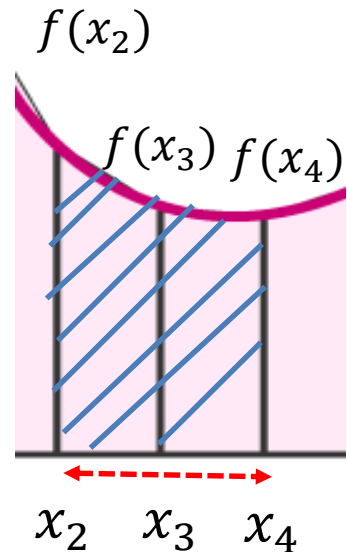
$$2h \frac{f(x_0) + 4f(x_1) + f(x_2)}{6}$$
$$= 0.8 \frac{0.2 + 4(1.288) + 2.456}{12}$$



Simpson's 1/3 Rule

- We first estimate the area from x_2 to x_4 .

- $x_2 = 0.4$ $f(0.4) = 2.456$
- $x_3 = 0.6$ $f(0.6) = 3.464$
- $x_4 = 0.8$ $f(0.8) = 0.232$



$$2h \frac{f(x_2) + 4f(x_3) + f(x_4)}{6}$$
$$= 0.8 \frac{2.456 + 4(3.464) + 0.232}{12}$$

Simpson's 1/3 Rule

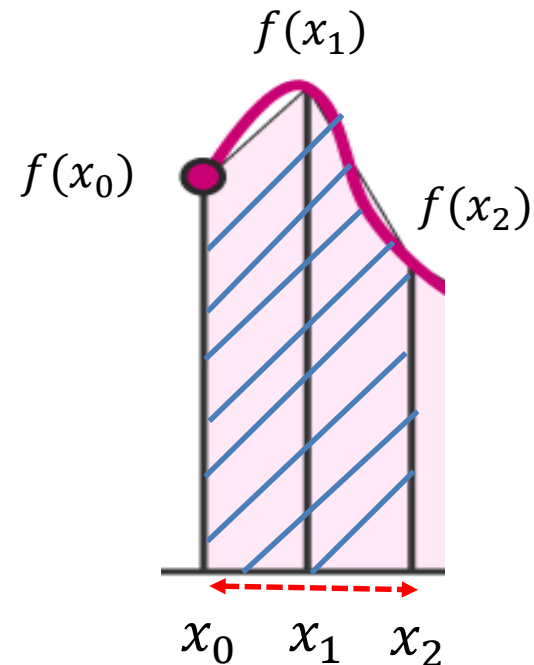
- The approximate value of the integral is

$$I = 0.8 \frac{0.2 + 4(1.288 + 3.464) + 2(2.456) + 0.232}{12} = 1.623467$$

- The error is $E_t = 1.640533 - 1.623467 = 0.017067$
- The percentage of the relative error is $\varepsilon_t = 1.04\%$
- Simpson's rule is **more accurate** than the trapezoidal rule. But, it requires **more computations**.

How does it work?

- Suppose that we need to **numerically compute** the **integral** of $f(x)$ over the interval $[x_0, x_2]$.
- We can **build an approximation of $f(x)$** using a **second-order Lagrange's polynomial $f_2(x)$** .



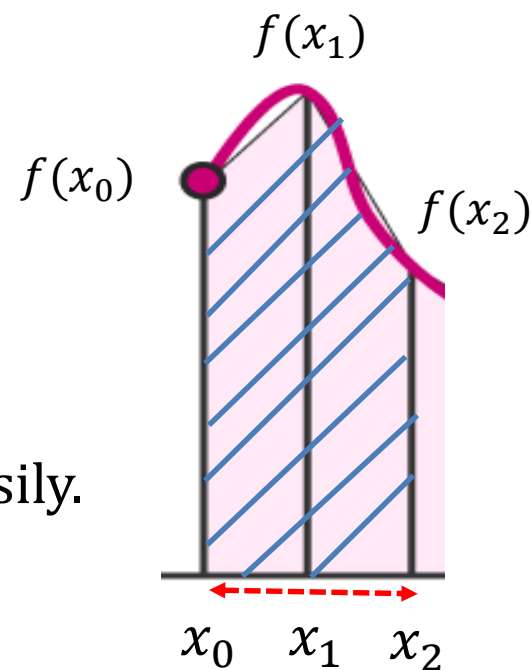
How does it work?

$$f_2(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)}f(x_0) + \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)}f(x_1)$$

$$+ \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}f(x_2)$$

Lagrange's approximation

$f_2(x)$ is a **polynomial**, we can compute its integral easily.



How does it work?

- We can then integrate $f_2(x)$ (**Lagrange's approximation**) instead of $f(x)$.

$$\begin{aligned} I &= \int_{x_0}^{x_2} \left[\frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} f(x_0) + \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} f(x_1) \right. \\ &\quad \left. + \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} f(x_2) \right] dx \\ &= 2h \frac{f(x_0) + 4f(x_1) + f(x_2)}{6} \end{aligned}$$

END

End 😊!